Multiplicative Normalizing Flows for Variational Bayesian Neural Networks

Christos Louizos [1 2] **Max Welling** [1 3]

Abstract

We reinterpret multiplicative noise in neural networks as auxiliary random variables that augment the approximate posterior in a variational setting for Bayesian neural networks. We show that through this interpretation it is both efficient and straightforward to improve the approximation by employing normalizing flows (Rezende & Mohamed, 2015) while still allowing for local reparametrizations (Kingma et al., 2015) and a tractable lower bound (Ranganath et al., 2015; Maaløe et al., 2016). In experiments we show that with this new approximation we can significantly improve upon classical mean field for Bayesian neural networks on both predictive accuracy as well as predictive uncertainty.

1. Introduction

Neural networks have been the driving force behind the success of deep learning applications. Given enough training data they are able to robustly model input-output relationships and as a result provide high predictive accuracy. However, they do have some drawbacks. In the absence of enough data they tend to overfit considerably; this restricts them from being applied in scenarios were labeled data are scarce, e.g. in medical applications such as MRI classification. Even more importantly, deep neural networks trained with maximum likelihood or MAP procedures tend to be overconfident and as a result do not provide accurate confidence intervals, particularly for inputs that are far from the training data distribution. A simple example can be seen at Figure 1a; the predictive distribution becomes overly overconfident, i.e. assigns a high softmax probability, towards the wrong class for things it hasn't seen before (e.g. an MNIST 3 rotated by 90 degrees). This in effect makes them unsuitable for applications where decisions are made, e.g. when a doctor determines the disease of a patient based on the output of such a network.

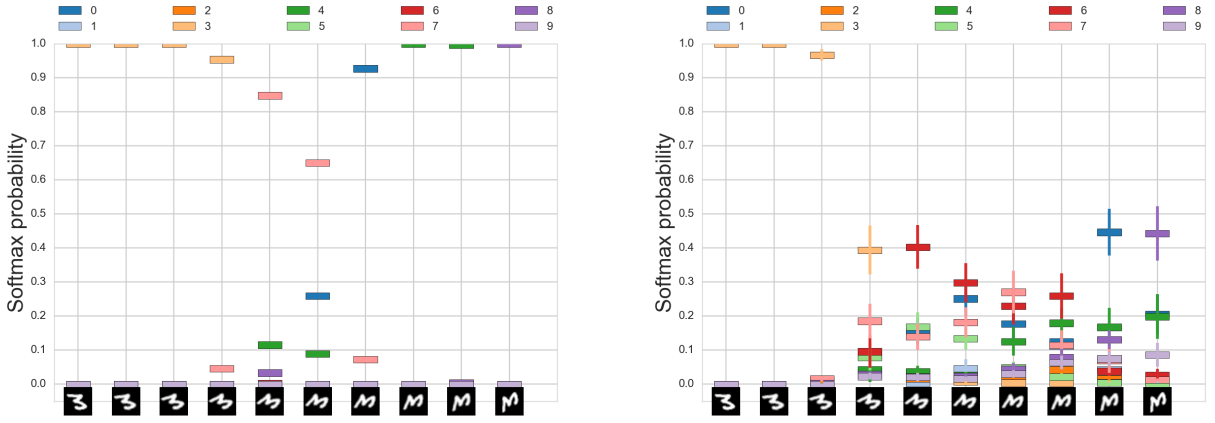
A principled approach to address both of the aforementioned shortcomings is through a Bayesian inference procedure. Under this framework instead of doing a point estimate for the network parameters we infer a posterior distribution. These distributions capture the parameter uncertainty of the network, and by subsequently integrating over them we can obtain better uncertainties about the predictions of the model. We can see that this is indeed the case at Figure 1b; the confidence of the network for the unseen digits is drastically reduced when we are using a Bayesian model, thus resulting into more realistic predictive distributions. Obtaining the posterior distributions is however no easy task, as the nonlinear nature of neural networks makes the problem intractable. For this reason approximations have to be made.

Many works have considered the task of approximate Bayesian inference for neural networks using either Markov Chain Monte Carlo (MCMC) with Hamiltonian Dynamics (Neal, 1995), distilling SGD with Langevin Dynamics (Welling & Teh, 2011; Korattikara et al., 2015) or deterministic techniques such as the Laplace Approximation (MacKay, 1992), Expectation Propagation (Hernández-Lobato & Adams, 2015; Hernández-Lobato et al., 2015) and variational inference (Graves, 2011; Blundell et al., 2015; Kingma et al., 2015; Gal & Ghahramani, 2015b; Louizos & Welling, 2016).

In this paper we will also tackle the problem of Bayesian inference in neural networks. We will adopt a stochastic gradient variational inference (Kingma & Welling, 2014; Rezende et al., 2014) procedure in order to estimate the posterior distribution over the weight matrices of the network. Arguably one of the most important ingredients of variational inference is the flexibility of the approximate posterior distribution; it determines how well we are able to capture the true posterior distribution and thus the true uncertainty of our models. In Section 2 we will show how we can produce very flexible distributions in an efficient way by employing auxiliary random variables (Agakov & Barber, 2004; Salimans et al., 2013; Ranganath et al., 2015; Maaløe et al., 2016) and normalizing flows (Rezende & Mohamed, 2015). In Section 3 we will discuss related

[1]University of Amsterdam, Netherlands [2]TNO Intelligent Imaging, Netherlands [3]Canadian Institute For Advanced Research (CIFAR). Correspondence to: Christos Louizos <c.louizos@uva.nl>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.



(a) LeNet with weight decay

(b) LeNet with multiplicative formalizing flows

Figure 1. Predictive distribution for a continuously rotated version of a 3 from MNIST. Each colour corresponds to a different class and the height of the bar denotes the probability assigned to that particular class by the network. Visualization inspired by (Gal & Ghahramani, 2015b).

work, whereas in Section 4 we will evaluate and discuss the proposed framework. Finally we will conclude with Section 5, where we will provide some final thoughts along with promising directions for future research.

2. Multiplicative normalizing flows

2.1. Variational inference for Bayesian Neural Networks

Let $\mathcal{D}$ be a dataset consisting of input output pairs $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ and let $\mathbf{W}_{1:L}$ denote the weight matrices of L layers. Assuming that $p(\mathbf{W}_i)$, $q_\phi(\mathbf{W}_i)$ are the prior and approximate posterior over the parameters of the i'th layer we can derive the following lower bound on the marginal log-likelihood of the dataset $\mathcal{D}$ using variational Bayes (Peterson, 1987; Hinton & Van Camp, 1993; Graves, 2011; Blundell et al., 2015; Kingma et al., 2015; Gal & Ghahramani, 2015b; Louizos & Welling, 2016):

$$\begin{aligned}\mathcal{L}(\phi) = \mathbb{E}_{q_\phi(\mathbf{W}_{1:L})}\big[\log p(\mathbf{y}|\mathbf{x}, \mathbf{W}_{1:L}) + \\ + \log p(\mathbf{W}_{1:L}) - \log q_\phi(\mathbf{W}_{1:L})\big],\end{aligned} \tag{1}$$

where $\tilde{p}(\mathbf{x}, \mathbf{y})$ denotes the training data distribution and ϕ the parameters of the variational posterior. For continuous $q(\cdot)$ distributions that allow for the reparametrization trick (Kingma & Welling, 2014) or stochastic backpropagation (Rezende et al., 2014) we can reparametrize the random sampling from $q(\cdot)$ of the lower bound in terms of noise variables ϵ and deterministic functions $f(\phi, \epsilon)$:

$$\begin{aligned}\mathcal{L} = \mathbb{E}_{p(\epsilon)}\big[\log p(\mathbf{y}|\mathbf{x}, f(\phi, \epsilon)) + \\ + \log p(f(\phi, \epsilon)) - \log q_\phi(f(\phi, \epsilon))\big].\end{aligned} \tag{2}$$

This reparametrization allow us to treat approximate parameter posterior inference as a straightforward optimization problem that can be optimized with off-the-shelf (stochastic) gradient ascent techniques.

2.2. Improving the variational approximation

For Bayesian neural networks the most common family for the approximate posterior is that of mean field with independent Gaussian distributions for each weight. Despite the fact that this leads to a straightforward lower bound for optimization, the approximation capability is quite limiting; it corresponds to just a unimodal "bump" on the very high dimensional space of the parameters of the neural network. There have been attempts to improve upon this approximation with works such as (Gal & Ghahramani, 2015b) with mixtures of delta peaks and (Louizos & Welling, 2016) with matrix Gaussians that allow for non-trivial covariances among the weights. Nevertheless, both of the aforementioned methods are still, in a sense, limited; the true parameter posterior is more complex than delta peaks or correlated Gaussians.

There has been a lot of recent work on ways to improve the posterior approximation in latent variable models with normalizing flows (Rezende & Mohamed, 2015) and auxiliary random variables (Agakov & Barber, 2004; Salimans et al., 2013; Ranganath et al., 2015; Maaløe et al., 2016) being the most prominent. Briefly, a normalizing flow is constructed by introducing parametrized bijective transformations, with easy to compute Jacobians, to random variables with simple initial densities. By subsequently optimizing the parameters of the flow according to the lower bound they can significantly improve the posterior approximation. Auxiliary random variables instead construct more flexible distributions by introducing latent variables in the

posterior itself, thus defining the approximate posterior as a mixture of simple distributions.

Nevertheless, applying these ideas to the parameters in a neural network has not yet been explored. While it is straightforward to apply normalizing flows to a sample of the weight matrix from $q(\mathbf{W})$, this quickly becomes very expensive; for example with planar flows (Rezende & Mohamed, 2015) we will need two extra matrices for each step of the flow. Furthermore, by utilizing this procedure we also lose the benefits of local reparametrizations (Kingma et al., 2015; Louizos & Welling, 2016) which are possible with Gaussian approximate posteriors.

In order to simultaneously maintain the benefits of local reparametrizations and increase the flexibility of the approximate posteriors in a Bayesian neural network we will rely on auxiliary random variables (Agakov & Barber, 2004; Salimans et al., 2013; 2015; Ranganath et al., 2015; Maaløe et al., 2016); more specifically we will exploit the well known "multiplicative noise" concept, e.g. as in (Gaussian) Dropout (Srivastava et al., 2014), in neural networks and we will parametrize the approximate posterior with the following process:

$$\mathbf{z} \sim q_\phi(\mathbf{z}); \qquad \mathbf{W} \sim q_\phi(\mathbf{W}|\mathbf{z}), \tag{3}$$

where now the approximate posterior becomes a compound distribution, $q(\mathbf{W}) = \int q(\mathbf{W}|\mathbf{z})q(\mathbf{z})d\mathbf{z}$, with $\mathbf{z}$ being a vector of random variables distributed according to the mixing density $q(\mathbf{z})$. To allow for local reparametrizations we will parametrize the conditional distribution for the weights to be a fully factorized Gaussian. Therefore we assume the following form for the fully connected layers:

$$q_\phi(\mathbf{W}|\mathbf{z}) = \prod_{i=1}^{D_{in}} \prod_{j=1}^{D_{out}} \mathcal{N}(z_i\mu_{ij}, \sigma_{ij}^2), \tag{4}$$

where D_{in}, D_{out} is the input and output dimensionality, and the following form for the kernels in convolutional networks:

$$q_\phi(\mathbf{W}|\mathbf{z}) = \prod_{i=1}^{D_h} \prod_{j=1}^{D_w} \prod_{k=1}^{D_f} \mathcal{N}(z_k\mu_{ijk}, \sigma_{ijk}^2), \tag{5}$$

where D_h, D_w, D_f are the height, width and number of filters for each kernel. Note that we did not let $\mathbf{z}$ affect the variance of the Gaussian approximation; in a pilot study we found that this parametrization was prone to local optima due to large variance gradients, an effect also observed with the multiplicative parametrization of the Gaussian posterior (Kingma et al., 2015; Molchanov et al., 2017). We have now reduced the problem of increasing the flexibility of the approximate posterior over the weights $\mathbf{W}$ to that of increasing the flexibility of the mixing density $q(\mathbf{z})$. Since $\mathbf{z}$ is of much lower dimension, compared to $\mathbf{W}$, it is now straightforward to apply normalizing flows to $q(\mathbf{z})$; in this way we can significantly enhance our approximation and allow for e.g. multimodality and nonlinear dependencies between the elements of the weight matrix. This will in turn better capture the properties of the true posterior distribution, thus leading to better performance and predictive uncertainties. We will coin the term *multiplicative normalizing flows* (MNFs) for this family of approximate posteriors. Algorithms 1, 2 describe the forward pass using local reparametrizations for fully connected and convolutional layers with this type of approximate posterior.

Algorithm 1 Forward propagation for each fully connected layer h. $\mathbf{M}_w, \mathbf{\Sigma}_w$ are the means and variances of each layer, $\mathbf{H}$ is a minibatch of activations and NF$(\cdot)$ is the normalizing flow described at eq. 6. For the first layer we have that $\mathbf{H} = \mathbf{X}$ where $\mathbf{X}$ is the minibatch of inputs.

Require: $\mathbf{H}, \mathbf{M}_w, \mathbf{\Sigma}_w$
$\mathbf{Z}_0 \sim q(\mathbf{z}_0)$
$\mathbf{Z}_{T_f} = \text{NF}(\mathbf{Z}_0)$
$\mathbf{M}_h = (\mathbf{H} \odot \mathbf{Z}_{T_f})\mathbf{M}_w$
$\mathbf{V}_h = \mathbf{H}^2\mathbf{\Sigma}_w$
$\mathbf{E} \sim \mathcal{N}(0, 1)$
return $\mathbf{M}_h + \sqrt{\mathbf{V}_h} \odot \mathbf{E}$

Algorithm 2 Forward propagation for each convolutional layer h. N_f are the number of convolutional filters, $*$ is the convolution operator and we assume the [batch, height, width, feature maps] convention.

Require: $\mathbf{H}, \mathbf{M}_w, \mathbf{\Sigma}_w$
$\mathbf{z}_0 \sim q(\mathbf{z}_0)$
$\mathbf{z}_{T_f} = \text{NF}(\mathbf{z}_0)$
$\mathbf{M}_h = \mathbf{H} * (\mathbf{M}_w \odot \text{reshape}(\mathbf{z}_{T_f}, [1, 1, D_f]))$
$\mathbf{V}_h = \mathbf{H}^2 * \mathbf{\Sigma}_w$
$\mathbf{E} \sim \mathcal{N}(0, 1)$
return $\mathbf{M}_h + \sqrt{\mathbf{V}_h} \odot \mathbf{E}$

For the normalizing flow of $q(\mathbf{z})$ we will use the masked RealNVP (Dinh et al., 2016) using the numerically stable updates introduced in Inverse Autoregressive Flow (IAF) (Kingma et al., 2016):

$$\begin{aligned} \mathbf{m} \sim \text{Bern}(0.5); \qquad & \mathbf{h} = \tanh(f(\mathbf{m} \odot \mathbf{z}_t)) \\ \boldsymbol{\mu} = g(\mathbf{h}); \qquad & \boldsymbol{\sigma} = \sigma(k(\mathbf{h})) \\ \mathbf{z}_{t+1} = \mathbf{m} \odot \mathbf{z}_t + (1 - \mathbf{m}) \odot (\mathbf{z}_t \odot \boldsymbol{\sigma} + (1 - \boldsymbol{\sigma}) \odot \boldsymbol{\mu}) & \end{aligned} \tag{6}$$

$$\log \left| \frac{\partial \mathbf{z}_{t+1}}{\partial \mathbf{z}_t} \right| = (1 - \mathbf{m})^T \log \boldsymbol{\sigma},$$

where $\odot$ corresponds to element-wise multiplication, $\sigma(\cdot)$

is the sigmoid function[1] and $f(\cdot), g(\cdot), k(\cdot)$ are linear mappings. We resampled the mask $\mathbf{m}$ every time in order to avoid a specific splitting over the dimensions of $\mathbf{z}$. For the starting point of the flow $q(\mathbf{z}_0)$ we used a simple fully factorized Gaussian and we will refer to the final iterate as $\mathbf{z}_{T_f}$.

2.3. Bounding the entropy

Unfortunately, parametrizing the posterior distribution as eq. 3 makes the lower bound intractable as generally we do not have a closed form density function for $q(\mathbf{W})$. This makes the calculation of the entropy $-\mathbb{E}_{q(\mathbf{W})}[\log q(\mathbf{W})]$ challenging. Fortunately we can make the lower bound tractable again by further lower bounding the entropy in terms of an auxiliary distribution $r(\mathbf{z}|\mathbf{W})$ (Agakov & Barber, 2004; Salimans et al., 2013; 2015; Ranganath et al., 2015; Maaløe et al., 2016). This can be seen as if we are performing variational inference on the augmented probability space $p(\mathcal{D}, \mathbf{W}_{1:L}, \mathbf{z}_{1:L})$, that maintains the same true posterior distribution $p(\mathbf{W}|\mathcal{D})$ (as we can always marginalize out $r(\mathbf{z}|\mathbf{W})$ to obtain the original model). The lower bound in this case becomes:

$$\begin{aligned}\mathcal{L}(\phi,\theta) = \mathbb{E}_{q_\phi(\mathbf{z}_{1:L},\mathbf{W}_{1:L})}\big[&\log p(\mathbf{y}|\mathbf{x},\mathbf{W}_{1:L},\mathbf{z}_{1:L}) + \\ &+ \log p(\mathbf{W}_{1:L}) + \log r_\theta(\mathbf{z}_{1:L}|\mathbf{W}_{1:L}) - \\ &- \log q_\phi(\mathbf{W}_{1:L}|\mathbf{z}_{1:L}) - \log q_\phi(\mathbf{z}_{1:L})\big],\end{aligned} \tag{7}$$

where θ are the parameters of the auxiliary distribution $r(\cdot)$. This bound is looser than the previous bound, however the extra flexibility of $q(\mathbf{W})$ can compensate and allow for a tighter bound. Furthermore, the tightness of the bound also depends on the ability of $r(\mathbf{z}|\mathbf{W})$ to approximate the "auxiliary" posterior distribution $q(\mathbf{z}|\mathbf{W}) = \frac{q(\mathbf{W}|\mathbf{z})q(\mathbf{z})}{q(\mathbf{W})}$. Therefore, to allow for a flexible $r(\mathbf{z}|\mathbf{W})$ we will follow (Ranganath et al., 2015) and we will parametrize it with inverse normalizing flows as follows:

$$r(\mathbf{z}_{T_b}|\mathbf{W}) = \prod_{i=1}^{D_z} \mathcal{N}(\tilde{\mu}_i, \tilde{\sigma}_i^2), \tag{8}$$

where for fully connected layers we have that:

$$\tilde{\mu}_i = \big(\mathbf{b}_1 \otimes \tanh(\mathbf{c}^T\mathbf{W})\big)(\mathbf{1} \odot D_{out}^{-1}) \tag{9}$$

$$\tilde{\sigma}_i = \sigma\Big(\big(\mathbf{b}_2 \otimes \tanh(\mathbf{c}^T\mathbf{W})\big)(\mathbf{1} \odot D_{out}^{-1})\Big), \tag{10}$$

and for convolutional:

$$\tilde{\mu}_i = \big(\tanh(\mathrm{mat}(\mathbf{W})\mathbf{c}) \otimes \mathbf{b}_1\big)(\mathbf{1} \odot (D_h D_w)^{-1}) \tag{11}$$

$$\tilde{\sigma}_i = \sigma\Big(\big(\tanh(\mathrm{mat}(\mathbf{W})\mathbf{c}) \otimes \mathbf{b}_2\big)(\mathbf{1} \odot (D_h D_w)^{-1})\Big), \tag{12}$$

where $\mathbf{b}_1, \mathbf{b}_2, \mathbf{c}$ are trainable vectors that have the same dimensionality as $\mathbf{z}$, D_z, $\mathbf{1}$ corresponds to a vector of 1s, $\otimes$ corresponds to the outer product and $\mathrm{mat}(\cdot)$ corresponds to the matricization[2] operator. The $\mathbf{z}_{T_b}$ variable corresponds to the fully factorized variable that is transformed by a normalizing flow to $\mathbf{z}_{T_f}$ or else the variable obtained by the inverse normalizing flow, $\mathbf{z}_{T_b} = \mathrm{NF}^{-1}(\mathbf{z}_{T_f})$. We will parametrize this inverse directly with the procedure described at eq. 6. Notice that we can employ local reparametrizations also in eq. 9,10,11,12, so as to avoid sampling the, potentially big, matrix $\mathbf{W}$. With the standard normal prior and the fully factorized Gaussian posterior of eq. 4 the KL-divergence between the prior and the posterior can be computed as follows:

$$\begin{aligned}&- KL(q(\mathbf{W})||p(\mathbf{W})) = \\ &\quad = \mathbb{E}_{q(\mathbf{W},\mathbf{z}_T)}[-KL(q(\mathbf{W}|\mathbf{z}_{T_f})||p(\mathbf{W})) + \\ &\quad + \log r(\mathbf{z}_{T_f}|\mathbf{W}) - \log q(\mathbf{z}_{T_f})],\end{aligned} \tag{13}$$

where each of the terms corresponds to:

$$\begin{aligned}&- KL(q(\mathbf{W}|\mathbf{z}_{T_f})||p(\mathbf{W})) = \\ &\quad = \frac{1}{2}\sum_{i,j}(-\log\sigma_{i,j}^2 + \sigma_{i,j}^2 + z_{T_{f_i}}^2\mu_{i,j}^2 - 1)\end{aligned} \tag{14}$$

$$\log r(\mathbf{z}_{T_f}|\mathbf{W}) = \log r(\mathbf{z}_{T_b}|\mathbf{W}) + \sum_{t=T_f}^{T_f+T_b} \log\left|\frac{\partial \mathbf{z}_{t+1}}{\partial \mathbf{z}_t}\right| \tag{15}$$

$$\log q(\mathbf{z}_{T_f}) = \log q(\mathbf{z}_0) - \sum_{t=1}^{T_f} \log\left|\frac{\partial \mathbf{z}_{t+1}}{\partial \mathbf{z}_t}\right|. \tag{16}$$

It should be noted that this bound is a generalization of the bound proposed by (Gal & Ghahramani, 2015b). We can arrive at the bound of (Gal & Ghahramani, 2015b) if we trivially parametrize the auxiliary model $r(\mathbf{z}|\mathbf{W}) = q(\mathbf{z})$ (which provides a less tight bound (Ranganath et al., 2015)) use a standard normal prior for $\mathbf{W}$, a Bernoulli $q(\mathbf{z})$ with probability of success π and then let the variance of our conditional Gaussian $q(\mathbf{W}|\mathbf{z})$ go to zero. This will result into the lower bound being infinite due to the log of the variances; nevertheless since we are not optimizing over σ we can simply disregard those terms. After a little bit of algebra we can show that the only term that will remain in the KL-divergence between $q(\mathbf{W})$ and $p(\mathbf{W})$ will be the expectation of the trace of the square of the mean matrix[3], i.e. $\mathbb{E}_{q(\mathbf{z})}[\frac{1}{2}\operatorname{tr}((\mathrm{diag}(\mathbf{z})\mathbf{M}))^T(\mathrm{diag}(\mathbf{z})\mathbf{M}))] = \frac{\pi}{2}\|\mathbf{M}\|_2^2$, with $1-\pi$ being the dropout rate.

We also found that in general it is beneficial to "constrain" the standard deviations σ_{ij} of the conditional Gaussian posterior $q(\mathbf{W}|\mathbf{z})$ during the forward pass for the computation

[1] $f(x) = \frac{1}{1+\exp(-x)}$

[2] Converting the multidimensional tensor to a matrix.

[3] The matrix that has $\mathbf{M}[i,j] = \mu_{ij}$

of the likelihood to a lower than the true range, e.g. $[0, \alpha]$ instead of the $[0, 1]$ we have with a standard normal prior. This results into a small bias and a looser lower bound, however it helps in avoiding bad local minima in the variational objective. This is akin to the free bits objective described at (Kingma et al., 2016).

3. Related work

Approximate inference for Bayesian neural networks has been pioneered by (MacKay, 1992) and (Neal, 1995). Laplace approximation (MacKay, 1992) provides a deterministic approximation to the posterior that is easy to obtain; it is a Gaussian centered at the MAP estimate of the parameters with a covariance determined by the inverse of the Hessian of the log-likelihood. Despite the fact that it is straightforward to implement, its scalability is limited unless approximations are made, which generally reduces performance. Hamiltonian Monte Carlo (Neal, 1995) is so far the golden standard for approximate Bayesian inference; nevertheless it is also not scalable to large networks and datasets due to the fact that we have to explicitly store the samples from the posterior. Furthermore as it is an MCMC method, assessing convergence is non trivial. Nevertheless there is interesting work that tries to improve upon those issues with stochastic gradient MCMC (Chen et al.) and distillation methods (Korattikara et al., 2015).

Deterministic methods for approximate inference in Bayesian neural networks have recently attained much attention. One of the first applications of variational inference in neural networks was in (Peterson, 1987) and (Hinton & Van Camp, 1993). More recently (Graves, 2011) proposed a practical method for variational inference in this setting with a simple (but biased) estimator for a fully factorized posterior distribution. (Blundell et al., 2015) improved upon this work with the unbiased estimator from (Kingma & Welling, 2014) and a scale mixture prior. (Hernández-Lobato & Adams, 2015) proposed to use Expectation Propagation (Minka, 2001) with fully factorized posteriors and showed good results on regression tasks. (Kingma et al., 2015) showed how Gaussian dropout can be interpreted as performing approximate inference with log-uniform priors, multiplicative Gaussian posteriors and local reparametrizations, thus allowing straightforward learning of the dropout rates. Similarly (Gal & Ghahramani, 2015b) showed interesting connections between Bernoulli Dropout (Srivastava et al., 2014) networks and approximate Bayesian inference in deep Gaussian Processes (Damianou & Lawrence, 2013) thus allowing the extraction of uncertainties in a principled way. Similarly (Louizos & Welling, 2016) arrived at the same result through structured posterior approximations via matrix Gaussians and local reparametrizations (Kingma et al., 2015).

It should also be mentioned that uncertainty estimation in neural networks can also be performed without the Bayesian paradigm; frequentist methods such as Bootstrap (Osband et al., 2016) and ensembles (Lakshminarayanan et al., 2016) have shown that in certain scenarios they can provide reasonable confidence intervals.

4. Experiments

All of the experiments were coded in Tensorflow (Abadi et al., 2016) and optimization was done with Adam (Kingma & Ba, 2015) using the default hyperparameters. We used the LeNet 5[4] (LeCun et al., 1998) convolutional architecture with ReLU (Nair & Hinton, 2010) nonlinearities. The means $\mathbf{M}$ of the conditional Gaussian $q(\mathbf{W}|\mathbf{z})$ were initialized with the scheme proposed in (He et al., 2015), whereas the log of the variances were initialized by sampling from $\mathcal{N}(-9, 0.001)$. Unless explicitly mentioned otherwise we use flows of length two for $q(\mathbf{z})$ and $r(\mathbf{z}|\mathbf{W})$ with 50 hidden units for each step of the flow of $q(\mathbf{z})$ and 100 hidden units for each step of the flow of $r(\mathbf{z}|\mathbf{W})$. We used 100 posterior samples to estimate the predictive distribution for all of the models during testing and 1 posterior sample during training.

Table 1. Models considered in this paper. Dropout corresponds to the model used in (Gal & Ghahramani, 2015a), Deep Ensemble to the model used in (Lakshminarayanan et al., 2016), FFG to the Bayesian neural network employed in (Blundell et al., 2015), FFLU to the Bayesian neural network used in (Kingma et al., 2015; Molchanov et al., 2017) with the additive parametrization of (Molchanov et al., 2017) and MNFG corresponds to the proposed variational approximation. It should be noted that Deep Ensembles use adversarial training (Goodfellow et al., 2014).

Name	Prior	Posterior
L2	$\mathcal{N}(\mathbf{0}, \mathbf{I})$	delta peak
Dropout	$\mathcal{N}(\mathbf{0}, \mathbf{I})$	mixture of zero and delta peaks
D. Ensem.	-	mixture of peaks
FFG	$\mathcal{N}(\mathbf{0}, \mathbf{I})$	fully factorized additive Gaussian
FFLU	$\log(\lvert\mathbf{W}\rvert) = \text{c}$	fully factorized additive Gaussian
MNFG	$\mathcal{N}(\mathbf{0}, \mathbf{I})$	multiplicative normalizing flows

4.1. Predictive performance and uncertainty

MNIST We trained on MNIST LeNet architectures using the priors and posteriors described at Table 1. We trained Dropout with the way described at (Gal & Ghahramani, 2015a) using 0.5 for the dropout rate and for Deep Ensembles (Lakshminarayanan et al., 2016) we used 10 members and $\epsilon = .25$ for the adversarial example generation. For the models with the Gaussian prior we constrained the standard deviation of the conditional posterior to be $\leq .5$

[4]The version from Caffe.

during the forward pass. The classification performance of each model can be seen at Table 2; while our overall focus is not classification accuracy per se, we see that with the MNF posteriors we improve upon mean field reaching similar accuracies with Deep Ensembles.

notMNIST To evaluate the predictive uncertainties of each model we performed the task described at (Lakshminarayanan et al., 2016); we estimated the entropy of the predictive distributions on notMNIST[5] from the LeNet architectures trained on MNIST. Since we a-priori know that none of the notMNIST classes correspond to a trained class (since they are letters and not digits) the ideal predictive distribution is uniform over the MNIST digits, i.e. a maximum entropy distribution. Contrary to (Lakshminarayanan et al., 2016) we do not plot the histogram of the entropies across the images but we instead use the empirical CDF, which we think is more informative. Curves that are closer to the bottom right part of the plot are preferable, as it denotes that the probability of observing a high confidence prediction is low. At Figure 2 we show the empirical CDF over the range of possible entropies, $[0, 2.5]$, for all of the models.

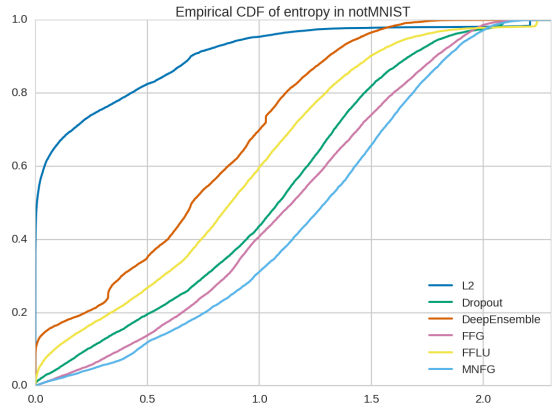


Figure 2. Empirical CDF for the entropy of the predictive distributions on notMNIST.

It is clear from the plot that the uncertainty estimates from MNFs are better than the other approaches, since the probability of a low entropy prediction is overall lower. The network trained with just weight decay was, as expected, the most overconfident with an almost zero median entropy while Dropout seems to be in the middle ground. The Bayesian neural net with the log-uniform prior also showed overconfidence in this task; we hypothesize that this is due to the induced sparsity (Molchanov et al., 2017) which results into the pruning of almost all irrelevant sources of variation in the parameters thus not providing enough variability to allow for uncertainty in the predictions. The sparsity levels[6] are 62%, 95.2% for the two convolutional layers and 99.5%, 93.3% for the two fully connected. Similar effects would probably be also observed if we optimized the dropout rates for Dropout. The only source of randomness in the neural network is from the Bernoulli random variables (r.v.) z. By employing the Central Limit Theorem[7] we can express the distribution of the activations as a Gaussian (Wang & Manning, 2013) with variance affected by the variance of the Bernoulli r.v., $\mathbb{V}(z) = \pi(1-\pi)$. The maximum variance of the Bernoulli r.v. is when $\pi = 0.5$, therefore any tuning of the Dropout rate will result into a decrease in the variance of the r.v. and therefore a decrease in the variance of the Gaussian at the hidden units. This will subsequently lead into less predictive variance and more confidence.

Finally, whereas it was shown at (Lakshminarayanan et al., 2016) that Deep Ensembles provide good uncertainty estimates (better than Dropout) on this task using fully connected networks, this result did not seem to apply for the LeNet architecture we considered. We hypothesize that they are sensitive to the hyperparameters (e.g. adversarial noise, number of members in the ensemble) and it requires more tuning in order to improve upon Dropout on this architecture.

CIFAR 10 We performed a similar experiment on CIFAR 10. To artificially create the "unobserved class" scenario, we hid 5 of the labels (dog, frog, horse, ship, truck) and trained on the rest (airplane, automobile, bird, cat, deer). For this task we used the larger LeNet architecture[8] described at (Gal & Ghahramani, 2015a). For the models with the Gaussian prior we similarly constrained the standard deviation during the forward pass to be $\leq .4$. For Deep Ensembles we used five members with $\epsilon = .1$ for the adversarial example generation. The predictive performance on these five classes can be seen in Table 2, with Dropout and MNFs achieving the overall better accuracies. We subsequently measured the entropy of the predictive distribution on the classes that were hidden, with the resulting empirical CDFs visualized in Figure 3.

We similarly observe that the network with just weight decay was the most overconfident. Furthermore, Deep Ensembles and Dropout had similar uncertainties, with Deep Ensembles having lower accuracy on the observed classes. The networks with the Gaussian priors also had similar uncertainty with the network with the log uniform prior, nevertheless the MNF posterior had much better accuracy on

[5]Can be found at http://yaroslavvb.blogspot.co.uk/2011/09/notmnist-dataset.html

[6]Computed by pruning weights where $\log \sigma^2 - \log \mu^2 \geq 5$ (Molchanov et al., 2017).

[7]Assuming that the network is wide enough.

[8]192 filters at each convolutional layer and 1000 hidden units for the fully connected layer.

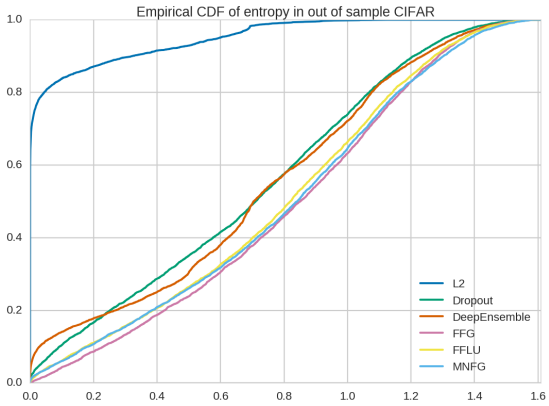


Figure 3. Empirical CDF for the entropy of the predictive distributions on the 5 hidden classes from CIFAR 10.

the observed classes. The sparsity levels for the network with the log-uniform prior now were 94.9%, 99.8% for the convolutional layers and 99.9%, 92.7% for the fully connected. Overall, the network with the MNF posteriors seem to provide the better trade-off in uncertainty and accuracy on the observed classes.

Table 2. Test errors (%) with the LeNet architecture on MNIST and the first five classes of CIFAR 10.

Dataset	L2	Dropout	D.Ensem.	FFG	FFLU	MNFG
MNIST	0.6	0.5	0.7	0.9	0.9	0.7
CIFAR 5	24	16	21	22	23	16

4.2. Accuracy and uncertainty on adversarial examples

We also measure how robust our models and uncertainties are against adversarial examples (Szegedy et al., 2013; Goodfellow et al., 2014) by generating examples using the fast sign method (Goodfellow et al., 2014) for each of the previously trained architectures using Cleverhans (Papernot et al., 2016). For this task we do not include Deep Ensembles as they are trained on adversarial examples.

MNIST On this scenario we observe interesting results if we plot the change in accuracy and entropy by varying the magnitude of the adversarial perturbation. The resulting plot can be seen in Figure 4. Overall Dropout seems to have better accuracies on adversarial examples; nevertheless, those come at an "overconfident" price since the entropy of the predictive distributions is quite low thus resulting into predictions that have, on average, above 0.7 probability for the dominant class. This is in contrast with MNFs; while the accuracy almost immediately drops close to random, the uncertainty simultaneously increases to almost maximum entropy. This implies that the predictive distribution is more or less uniform over those examples. So despite the fact that our model cannot overcome adversarial examples at least it "knows that it doesn't know".

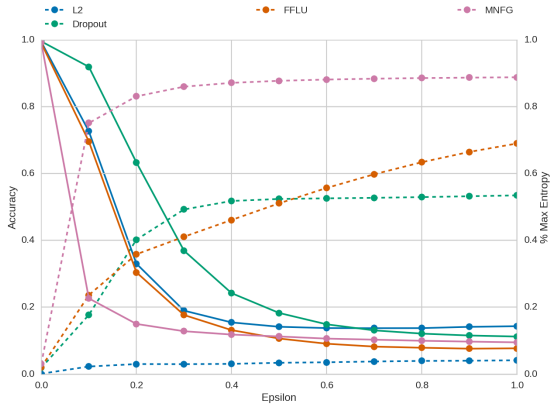


Figure 4. Accuracy (solid) vs entropy (dashed) as a function of the adversarial perturbation ϵ on MNIST.

CIFAR We performed the same experiment also on the five class subset of CIFAR 10. The results can be seen in Figure 5. Here we however observe a different picture, compared to MNIST, since all of the methods experienced overconfidence. We hypothesize that adversarial examples are harder to escape and be uncertain about in this dataset, due to the higher dimensionality, and therefore further investigation is needed.

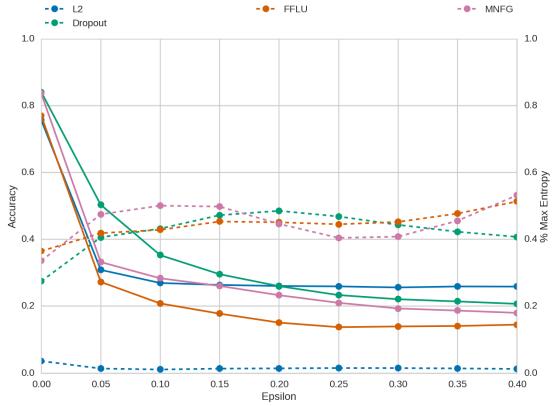
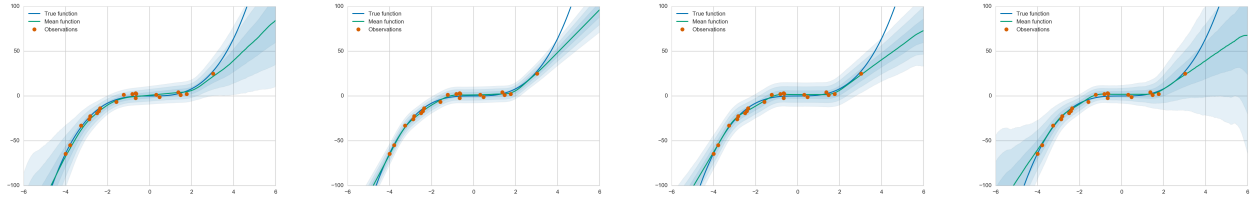


Figure 5. Accuracy (solid) vs entropy (dashed) as a function of the adversarial perturbation ϵ on CIFAR 10 (on the first 5 classes).

4.3. Regression on toy dataset

For the final experiment we visualize the predictive distributions obtained with the different models on the toy regression task introduced at (Hernández-Lobato & Adams, 2015). We generated 20 training inputs from $\mathcal{U}[-4, 4]$ and then obtained the corresponding targets via $y = x^3 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 9)$. We fixed the likelihood noise to its true value and then fitted a Dropout network with $\pi = 0.5$



(a) Dropout $\pi = 0.5$ (b) Dropout learned π (c) FFLU (d) MNFG

Figure 6. Predictive distributions for the toy dataset. Blue areas correspond to ± 3 standard deviations around the mean.

for the hidden layer[9], an FFLU network and an MNFG. We also fitted a Dropout network where we also learned the dropout probability π of the hidden layer according to the bound described at section 2.3 (which is equivalent to the one described at (Gal & Ghahramani, 2015b)) using REINFORCE (Williams, 1992) and a global baseline (Mnih & Gregor, 2014). The resulting predictive distributions can be seen at Figure 6.

As we can observe, MNF posteriors provide more realistic predictive distributions, closer to the true posterior (which can be seen at (Hernández-Lobato & Adams, 2015)) and with the network being more uncertain on areas where we do not observed any data. The uncertainties obtained by Dropout with fixed $\pi = 0.5$ did not diverge as much in those areas but overall they were better compared to the uncertainties obtained with FFLU. We could probably attribute the latter to the sparsification of the network since 95% and 44% of the parameters were pruned for each layer respectively.

Interestingly the uncertainties obtained with the network with the learned Dropout probability were the most "overfitted". This might suggest that Dropout uncertainty is probably not a good posterior approximation since by optimizing the dropout rates we do not seem to move closer to the true posterior predictive distribution. This is in contrast with MNFs; they are flexible enough to allow for optimizing all of their parameters in a way that does better approximate the true posterior distribution. This result also empirically verifies the claim we previously made; by learning the dropout rates the entropy of the posterior predictive will decrease thus resulting into more overconfident predictions.

5. Conclusion

We introduce multiplicative normalizing flows (MNFs); a family of approximate posteriors for the parameters of a variational Bayesian neural network. We have shown that through this approximation we can significantly improve upon mean field on both predictive performance as well as predictive uncertainty. We compared our uncertainty on notMNIST and CIFAR with Dropout (Srivastava et al., 2014; Gal & Ghahramani, 2015b) and Deep Ensembles (Lakshminarayanan et al., 2016) using convolutional architectures and found that MNFs achieve more realistic uncertainties while providing predictive capabilities on par with Dropout. We suspect that the predictive capabilities of MNFs can be further improved through more appropriate optimizers that avoid the bad local minima in the variational objective. Finally, we also highlighted limitations of Dropout approximations and empirically showed that MNFs can overcome them.

There are a couple of promising directions for future research. One avenue would be to explore how much can MNFs sparsify and compress neural networks under either sparsity inducing priors, such as the log-uniform prior (Kingma et al., 2015; Molchanov et al., 2017), or empirical priors (Ullrich et al., 2017). Another promising direction is that of designing better priors for Bayesian neural networks. For example (Neal, 1995) has identified limitations of Gaussian priors and proposes alternative priors such as the Cauchy. Furthermore, the prior over the parameters also affects the type of uncertainty we get in our predictions; for instance we observed in our experiments a significant difference in uncertainty between Gaussian and log-uniform priors. Since different problems require different types of uncertainty it makes sense to choose the prior accordingly, e.g. use an informative prior so as to alleviate adversarial examples.

[9]No Dropout was used for the input layer since it is 1-dimensional.

Acknowledgements

We would like to thank Klamer Schutte, Matthias Reisser and Karen Ullrich for valuable feedback. This research is supported by TNO, NWO and Google.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, et al. Tensorflow: Large-scale machine learning on heterogeneous

distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Agakov, Felix V and Barber, David. An auxiliary variational method. In *International Conference on Neural Information Processing*, pp. 561–566. Springer, 2004.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015.

Chen, Tianqi, Fox, Emily B, and Guestrin, Carlos. Stochastic gradient hamiltonian monte carlo.

Damianou, Andreas C. and Lawrence, Neil D. Deep gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, pp. 207–215, 2013.

Dinh, Laurent, Sohl-Dickstein, Jascha, and Bengio, Samy. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Gal, Yarin and Ghahramani, Zoubin. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015a.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015b.

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Graves, Alex. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

Hernández-Lobato, José Miguel and Adams, Ryan. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 1861–1869, 2015.

Hernández-Lobato, José Miguel, Li, Yingzhen, Hernández-Lobato, Daniel, Bui, Thang, and Turner, Richard E. Black-box α-divergence minimization. *arXiv preprint arXiv:1511.03243*, 2015.

Hinton, Geoffrey E and Van Camp, Drew. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13. ACM, 1993.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR), San Diego*, 2015.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

Kingma, Diederik P, Salimans, Tim, and Welling, Max. Variational dropout and the local reparametrization trick. *Advances in Neural Information Processing Systems*, 2015.

Kingma, Diederik P, Salimans, Tim, and Welling, Max. Improving variational inference with inverse autoregressive flow. *arXiv preprint arXiv:1606.04934*, 2016.

Korattikara, Anoop, Rathod, Vivek, Murphy, Kevin, and Welling, Max. Bayesian dark knowledge. *arXiv preprint arXiv:1506.04416*, 2015.

Lakshminarayanan, Balaji, Pritzel, Alexander, and Blundell, Charles. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Louizos, Christos and Welling, Max. Structured and efficient variational deep learning with matrix gaussian posteriors. *arXiv preprint arXiv:1603.04733*, 2016.

Maaløe, Lars, Sønderby, Casper Kaae, Sønderby, Søren Kaae, and Winther, Ole. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

MacKay, David JC. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3): 448–472, 1992.

Minka, Thomas P. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 362–369. Morgan Kaufmann Publishers Inc., 2001.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014.

Molchanov, D., Ashukha, A., and Vetrov, D. Variational Dropout Sparsifies Deep Neural Networks. *ArXiv e-prints*, January 2017.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

Neal, Radford M. *Bayesian learning for neural networks*. PhD thesis, Citeseer, 1995.

Osband, Ian, Blundell, Charles, Pritzel, Alexander, and Van Roy, Benjamin. Deep exploration via bootstrapped dqn. *arXiv preprint arXiv:1602.04621*, 2016.

Papernot, Nicolas, Goodfellow, Ian, Sheatsley, Ryan, Feinman, Reuben, and McDaniel, Patrick. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

Peterson, Carsten. A mean field theory learning algorithm for neural networks. *Complex systems*, 1:995–1019, 1987.

Ranganath, Rajesh, Tran, Dustin, and Blei, David M. Hierarchical variational models. *arXiv preprint arXiv:1511.02386*, 2015.

Rezende, Danilo Jimenez and Mohamed, Shakir. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1278–1286, 2014.

Salimans, Tim, Knowles, David A, et al. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

Salimans, Tim, Kingma, Diederik P, Welling, Max, et al. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pp. 1218–1226, 2015.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Ullrich, Karen, Meeds, Edward, and Welling, Max. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

Wang, Sida and Manning, Christopher. Fast dropout training. In *Proceedings of The 30th International Conference on Machine Learning*, pp. 118–126, 2013.

Welling, Max and Teh, Yee W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 681–688, 2011.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

A. Memorization capabilities

As it was shown in (Zhang et al., 2016), deep neural networks can exhibit memorization, even with random labels. Therefore deep neural networks could perfectly fit the training data while having random chance accuracy on the test data, even with Dropout or weight decay regularization. (Molchanov et al., 2017) instead showed that by employing Sparse Variational Dropout this phenomenon did not appear, thus resulting into the network pruning everything and having random chance accuracy on both training and test sets. We similarly show here that with Gaussian priors and MNF posteriors we also have random chance accuracy on both train and test sets. This suggests that it is proper Bayesian inference that penalizes memorization.

Table 3. Accuracy (%) with the LeNet architecture on MNIST and the first five classes of CIFAR 10 using random labels. Random chance is 11% on MNIST and 20% on CIFAR 5.

Dataset	**Dropout** train	**Dropout** test	**MNFG** train	**MNFG** test
MNIST	30	11	11	11
CIFAR 5	89	20	20	20